Leakage-Aware Cooling Management for Improving Server Energy Efficiency

Marina Zapater, Ozan Tuncer, José L. Ayala, José M. Moya, Kalyan Vaidyanathan, Kenny Gross and Ayse K. Coskun

Abstract—The computational and cooling power demands of enterprise servers are increasing at an unsustainable rate. Understanding the relationship between computational power, temperature, leakage, and cooling power is crucial to enable energy-efficient operation at the server and data center levels. This paper develops empirical models to estimate the contributions of static and dynamic power consumption in enterprise servers for a wide range of workloads, and analyzes the interactions between temperature, leakage, and cooling power for various workload allocation policies. We propose a cooling management policy that minimizes the server energy consumption by setting the optimum fan speed during runtime. Our experimental results on a presently shipping enterprise server demonstrate that including leakage awareness in workload and cooling management provides additional energy savings without any impact on performance.

Keywords—Energy-efficiency, cooling control, leakage power, data centers

1 Introduction

Data centers often comprise thousands of enterprise servers that typically serve millions of users globally in a 24-7 fashion. The increasing demand for computing resources has recently facilitated the rapid proliferation and growth of data center facilities. Until recently, data centers have focused mainly on providing the desired performance. As a result, raw throughput increased tremendously. However, today's data centers consume a huge amount of electrical power. In 2010, data center electricity represented 1.3% of all the electricity use in the world, yielding 250 billion kWh consumption per year worldwide [1]. In year 2012 alone, global data center power consumption increased to 38GW. A further rise of 17% to 43GW was estimated in 2013 [2].

The cooling power needed to keep the servers within reliable thermal operating conditions has traditionally been one of the major contributors to the overall data center power consumption, accounting for over 30% of the electricity bill [3]. In the last years, significant research effort has been devoted to decrease the cooling power, thus increasing the data center Power Usage Effectiveness (PUE), defined as the ratio between total facility power and IT power. According to a report by the Uptime Institute, average PUE improved from 2.5 in 2007 to 1.89 in 2012, reaching 1.65 in 2013 [4]. This average PUE values are still far from the 1.1 to 1.3 obtained in data centers using the most efficient free cooling techniques [5], that allow to reach values as low as the 1.13 achieved by Google Data Centers [6]. Apart from using more efficient room cooling systems, raising the inlet temperature is one of the most common strategies to increase efficiency [4]. The increase in room ambient temperature, however, also increases the fan speed of servers to keep all components below critical thermal thresholds. As fan power is a cubic function of fan speed, using high fan speeds leads to a high cumulative server fan power. Fans have become an important contributor to power consumption, reaching up to 14% of the overall data center power consumption [7].

Higher room temperatures also imply increasing the chip temperatures, which are already high due to the rapid increase in CMOS power density [8]. This may cause potential reliability problems as well as increased leakage power because of the exponential dependence of leakage on temperature. Prior work analyzing the effect of leakage on servers highlights that allowing higher room temperatures may or may not be efficient depending on the specific data center configuration [9].

Another major factor that affects temperature in servers is the workload dynamics. Different workload allocation schemes change the temperature balance across the chip and thus, the leakage power [10]. Moreover, server power consumption depends on the characteristics of the running workload and the allocation policy [11]. State-of-the-art techniques are either focused at the CPU level, or, if scaled to the server level, they tackle

fan control, leakage power reduction, and temperature-aware workload allocation problems separately [12]. However, server temperature and energy depend on decisions in all these domains. In order to obtain the highest possible energy savings in the overall server power consumption, the dependencies between these domains need to be considered, motivating the design of a comprehensive multivariate control strategy.

This paper proposes a strategy to reduce server energy consumption, in a way that is aware of the interactions among power, temperature, leakage, and workload dynamics. Our specific contributions are as follows:

- We design empirical models to estimate various power components in the server (e.g., static and dynamic power, CPU and memory power). We validate our models using a wide range of applications running on a presently-shipping enterprise server.
- We analyze leakage vs. cooling power tradeoffs at the server level, and show the importance of temperature-dependent leakage in server energy consumption. We also study the relationship among power, temperature, application characteristics and workload allocation.
- We develop a control strategy that dynamically sets the optimum cooling for arbitrary workloads. Compared to prior techniques [13], [14], our policy reduces leakage plus fan energy by an additional 3% without any impact on performance.

The rest of this paper starts by discussing the related work. Section 3 shows our experimental methodology. The proposed power modeling techniques are presented in Section 4. Section 5 describes the proposed fan control policy. In Section 6, we analyze the impact of workload allocation on energy. Results are presented in Section 7, and Section 8 concludes the paper.

2 Related work

In the area of server energy efficiency, several works tackle fan control to reduce cooling costs. Han et al. [12] propose a runtime fan controller based on offline thermal modeling validated via simulation. Shin et al. [15] use Dynamic Voltage-Frequency Scaling (DVFS) together with fan control to minimize cooling and CPU power in a desktop computer. Chan et al. [16] approach the fan control problem both from the energy minimization and fan-induced vibration perspective. Even though our work could be combined with DVFS, our goal is to minimize overall server energy without relying on this technique as it introduces penalties in execution time, potentially increasing energy consumption. Moreover, our work minimizes leakage and cooling power by proactively setting the optimum fan speed before a thermal event occurs, and is validated on an enterprise server.

Other approaches that take into account the leakage-cooling tradeoffs do not include a setup that enables fan speed control. Policies such as TAPO-server, proposed by Huang et al. [13], indirectly vary fan speed by controlling the processor thermal threshold at runtime to reactively find the optimum fan speed. TAPO is effective only with constant workloads as it waits for the thermal steady-state to control the fan speed. Similarly, recent work by Pradelle et.al. [17] uses a hill-climbing optimization technique that relies on utilization as a proxy variable for the estimation of heat dissipation which, as we show in this paper, is not sufficient to select the optimum cooling for an arbitrary workload. In our work, we have direct control over the cooling subsystem of the server. Moreover, to enable proactiveness, we develop power and thermal models of the server to predict the leakage and cooling power for arbitrary workloads.

Prior work on server power modeling usually focuses on estimating the dynamic power consumed by servers. Lewis et al. [18] develop a linear regression model based on performance counters to provide run-time system-wide power prediction. Other linear models formulate server power as a quadratic function of CPU usage [19]. The power modeling technique vMeter [20], observes a correlation between the total system power consumption and component utilization, and creates a linear total server power model. Cochran et al. [21] determine the relevant workload metrics for energy minimization and manage tradeoffs between energy and delay. Previous approaches in server power modeling assume that leakage has minimal impact and disregard cooling power. Our work, on the contrary, presents an accurate model for leakage power consumption and shows its impact on total power consumption, and is robust to changes in the workload allocation policy.

There are some recent techniques that consider fan control together with scheduling for multi-objective optimization [16], [22]. These approaches make use of a joint energy, thermal and cooling management technique to reduce the server cooling and memory energy costs. They propose a thermal model that uses electrical analogies to represent the thermal coupling between the server components and the effect of fan speed on heat dissipation. Our work, on the contrary, is able to split and separately quantify the contributions of cooling power from those of leakage and total system power.

To the best of our knowledge, our approach is the first to present a leakage-aware multivariate cooling management strategy that is robust to arbitrary workloads and allocation policies running on a presently-shipping enterprise server. Compared to our earlier work [14], this paper includes models that are robust and accurate for a wide range of real applications, a proactive runtime fan control policy, and an analysis of the leakage vs. cooling tradeoffs for various workload allocation scenarios.

3 Methodology

The main purposes of the server fans are to remove the heat produced and to prevent the overheating of the hottest components such as CPUs and memories. The fan speed should be carefully selected to avoid overcooling,

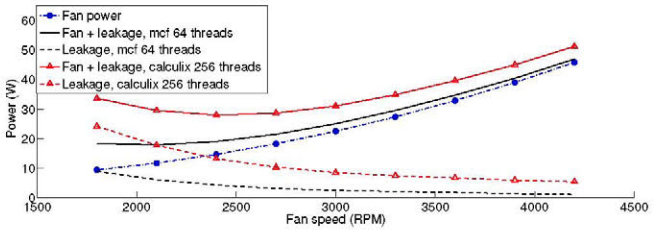

Fig. 1: Fan and leakage power for various workloads.

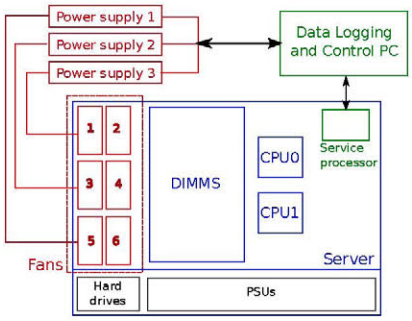

Fig. 2: Experimental setup and server internals diagram.

which implies high cooling costs, and also overheating, which results in shorter component lifetimes and higher leakage power. To clarify this point, Figure 1 shows the cubic increase in fan power with fan speed as well as the exponential increase in leakage power when fan speed decreases for two particular workloads running on a highly multi-threaded enterprise server: (i) a memory intensive workload utilizing 25% of the server (64 copies of *mcf*) and (ii) a CPU intensive workload fully utilizing the server (256 copies of *calculix*). We observe that different RPM settings minimize the total fan plus leakage power for the two workload scenarios.

The goal of our work is to reduce the energy consumption in enterprise servers found in energy-hungry data centers. To this end, we propose a proactive fan speed policy that sets the optimum cooling in a way that is aware of the leakage-cooling tradeoffs at the server, and yet robust to different workload allocation policies. To build this proactive policy, we need to develop accurate models that predict the leakage and cooling power. Therefore, an experimental setup is required to isolate and control the cooling subsystem, as well as to gather workload and sensor data from the server.

Experimental setup

All experiments are carried out in the above mentioned presently-shipping enterprise server, which contains two SPARC T3 processors [23] in 2 sockets that provide a total of 256 hardware threads, 32 8GB memory DIMMs, and 2 hard drives. We enable customized fan control by setting the fan currents through external Agilent E3644A power supplies, as shown in Figure 2.

We map the input current values to fan speeds, which are inferred with very high accuracy by taking the FFT of vibration sensors. In our work, we use a minimum fan speed of 1800RPM, and a maximum of 4200RPM. 1800RPM is sufficiently low to observe how leakage becomes dominant over fan power in our system (see Figure 1). Moreover, fan speeds lower than 1800RPM lead to unstable fan behavior. On the other hand, 4200RPM overcools the server under our experimental conditions, and is above the maximum server default fan speed.

The fan speed can be remotely adjusted by software scripts in the Data Logging and Control PC (DLC-PC), which also collects server sensor data through the Continuous System Telemetry Harness (CSTH) [24]. The CSTH runs in the service processor of the enterprise server as a part of the existing system software stack; therefore, no overhead is introduced by the sensor data processing. For our experiments, we collect the following sensor data: (i) CPU and memory temperature, (ii) per-CPU voltage and current, (iii) total server power. We poll the sensors every second to observe the power and thermal behavior with sufficient granularity.

We use Solaris 10 OS tools (*sar*, *cpustat*, *busstat* and *iostat*) to poll the hardware counters for workload characterization. The overhead introduced by polling the counters during execution is negligible.

In this paper, we run a comprehensive set of workloads to train and test our models and policies. For model training, we use two synthetic workloads that allow to stress different components of the system:

- *LoadGen* is a customized load-synthesis tool that obtains the highest possible gate switching in the chips and provides customized dynamic profiles that meet any desired utilization level.
- *RandMem* is a synthetic benchmark that accesses random memory regions of a given size with a given access pattern. The original benchmark [25] is modified to stress the large DRAMs in our system.

To validate the developed models and to evaluate our policies, we use: (i) SPEC Power_ssj2008 [26], a benchmark that evaluates the power and performance characteristics of volume class servers, (ii) a subset of floating point (FP) and integer workloads from the CPU-intensive SPEC CPU 2006 [27] benchmark that exhibit a distinctive set of characteristics [28], and (iii) the PARSEC multi-threaded benchmark suite [29] that assesses the performance of multiprocessor systems.

4 Modeling

This section presents the server power and temperature models needed to enable proactive cooling management. First, we model the temperature-dependent power consumption in the server. Among the enterprise server components, CPUs exhibit the majority of the temperature-dependent leakage power [9].

Apart from the leakage awareness, we also consider lowering the CPU temperature by achieving a flatter thermal profile via workload allocation. Changing the

workload allocation to the processor cores has an impact on both temperature and energy, affecting the power consumption of both CPU and memory. In order to reliably evaluate the impact of different allocation schemes, we also model the memory power and validate that memory power does not depend on temperature.

Finally, we develop a CPU temperature model which enables us to estimate the temperature attained by a certain workload and to proactively set the fan speed to the optimum cooling conditions.

4.1 Server power modeling

The power consumption of a server can be split into three different contributors: (i) the dynamic or active power, (ii) the static power, and (iii) the cooling power due to the server fans:

$$P_{server} = P_{static} + P_{dynamic} + P_{fan} \quad (1)$$

Static power consumption refers to the cumulative idle server power of all the server components and the temperature-dependent leakage power, whereas dynamic power is inherent to the execution of a certain workload. In our system, CSTH provides the overall power consumption (P_{server}) using sensor measurements, whereas the cooling power (P_{fan}) is isolated and can be measured independently.

We further divide P_{static} into two components as $P_{static} = P_{idle} + P_{leakT}$, where P_{idle} represents the idle power of all components when leakage is minimum, i.e., at the maximum server fan speed (4200RPM), and P_{leakT} is the temperature-dependent leakage power due to the increase in temperature during workload execution.

Similarly, we divide the workload induced dynamic power into its sub-components as follows:

$$P_{dynamic} = P_{CPU,dyn} + P_{mem,dyn} + P_{other,dyn} \quad (2)$$

where $P_{CPU,dyn}$ is the dynamic CPU power, $P_{mem,dyn}$ is the dynamic memory power, and P_{other} is the contribution of other components. This last component is mainly composed of disk and network activity. Although its absolute value can be significant in some workloads, P_{other} has negligible dependence on workload allocation and temperature for the workloads we run.

In order to find the optimum cooling conditions at runtime, we need to model the temperature-dependent leakage power P_{leakT}. Additionally, to analyze the impact of workload allocation, we need to derive a model for memory power. In the next subsections, we provide a detailed explanation on these models.

CPU power

As the temperature-dependent leakage is mainly due to CPU leakage, we develop an empirical CPU power model, and validate our assumption by observing that overall server leakage can be expressed by the CPU leakage with sufficient accuracy.

Equation 3 shows how CPU power can be divided into $P_{CPU,idle}$, which contains a temperature-independent leakage plus the power consumption due to the OS running, a temperature-dependent leakage component ($P_{CPU,leakT}$), and the dynamic power due to workload execution ($P_{CPU,dyn}$):

$$P_{CPU} = P_{CPU,idle} + P_{CPU,leakT} + P_{CPU,dyn} \quad (3)$$

As CSTH provides P_{CPU} and $P_{CPU,idle}$ using voltage/current sensor readings, we only need to model $P_{CPU,leakT}$ and $P_{CPU,dyn}$. We start by modeling the temperature-dependent leakage power, $P_{CPU,leakT}$.

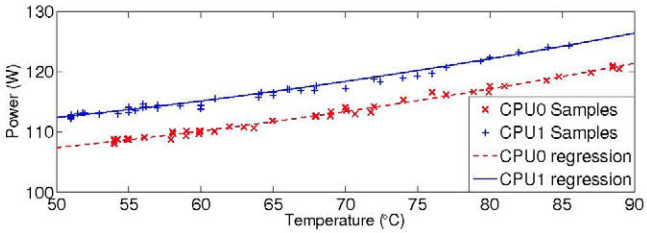

Fig. 3: Temperature-dependent CPU leakage model regression for both CPUs in the system.

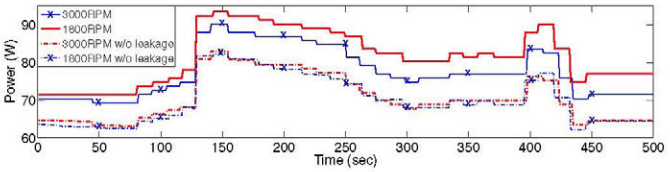

Fig. 4: Temperature-dependent CPU leakage model validation for 128 copies of mcf running on CPU0.

Temperature-dependent CPU leakage

To train this model, we use *LoadGen* synthetic workload with full utilization. We run the same workload under different fan speeds ranging from 1800RPM to 4200RPM, and measure CPU power and temperature. Because the workload is constant in all experiments and the only control knob is fan speed, power consumption can only change due to the temperature-dependent leakage. As leakage power depends exponentially on temperature, we use the measured CPU temperature and power to regress the Taylor series expansion of an exponential:

$$P_{leak} = \alpha_0 + \alpha_1 \cdot T_{CPU} + \alpha_2 \cdot T_{CPU}^2 \quad (4)$$

where α_i's are regression coefficients, and T_{CPU} is the CPU temperature in Celsius. We derive the above model for each of the two CPUs in our system.

Figure 3 shows the data regression against the measured samples of the training set. The model exhibits a Root-Mean-Square Error (RMSE) of 0.39W and 0.45W for CPU0 and CPU1, respectively, for the training set.

To validate our model, we run our test workloads under different fan speeds, and subtract $P_{CPU,leakT}$ from the power traces. Because the executions of a given workload only differ in fan speed, the remaining power ($P_{CPU,idle} + P_{CPU,dyn}$) should be the same. Figure 4

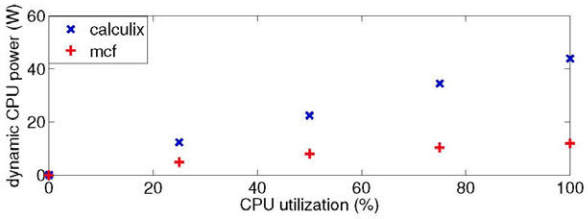

Fig. 5: Dynamic CPU power vs. utilization for selected SPEC CPU workloads.

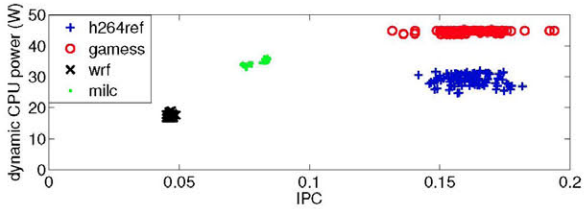

Fig. 6: Dynamic CPU power vs. IPC for 128 concurrent copies of selected SPEC CPU workloads.

shows two example traces of our validation using two different fan speeds. The difference between the curves once leakage has been subtracted is a direct analytical estimate of the error of our model. We apply the aforementioned methodology to all the SPEC CPU and PARSEC workloads in our test set (*mcf, sjeng, libquantum, cactusADM, zeusmp, lbm, calculix* from SPEC CPU 2006, and *fluidanimate, canneal, bodytrack, streamcluster, ferret, facesim* from PARSEC) when running with 64, 128 and 192 threads, and compute the difference between the resultant curves. The average error in the test set is only 0.67W, which shows very high accuracy.

Dynamic CPU power

Finally, to model the dynamic CPU power, prior work suggests using utilization [17] or number of retired instructions per cycle (IPC) [30]. However, Figure 5 clearly shows that the utilization is not a reliable metric for modeling power in our hyper-threaded multi-core processor, as the same utilization value can correspond to important differences in dynamic CPU power. Similarly, as can be observed in Figure 6, IPC is also an inaccurate power metric as the same IPC value can correspond to different dynamic CPU power levels. Because of these outcomes, our dynamic power prediction is based on our leakage model. We directly subtract the estimated leakage power and idle power from the measured CPU power to obtain the dynamic power using Equation (3).

Memory power

This section presents our memory power model, which is used both to confirm that the memory power does not depend on temperature and to explain the impact of workload allocation on server power.

We use a modified version of the synthetic benchmark *RandMem* to train our memory model. *RandMem* stresses

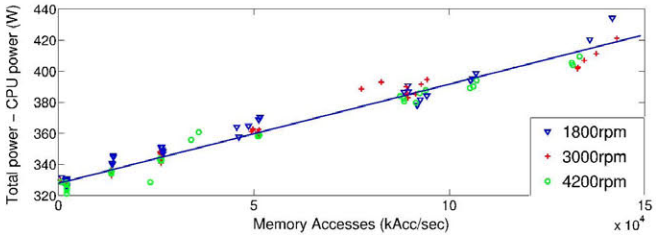

Fig. 7: Server power vs. number of memory accesses for RandMem workload under different fan speeds

the memory with desired number of read-write accesses using a memory space from 512Mb to 64GB.

In our system, we have two available power measurements: CPU voltage/current sensors that allow measuring P_{CPU}, and power sensors that measure P_{server}. As the benchmark *RandMem* has negligible disk and network number of accesses, we directly use the power difference $P_{server} - P_{CPU}$ to train the model.

Figure 7 shows how memory power grows linearly with the number of memory accesses per second. We experiment with three different fan speeds to test whether the memory power depends on temperature. As seen in the figure, samples representing different fan speeds are distributed along the plot, indicating that there is no significant dependence between memory power and temperature. Hence, we conclude that the temperature-dependent leakage power is mostly explained by the CPU leakage, agreeing with prior work [9]. Based on this observation, we use Equation (5) to model memory power consumption:

$$P_{mem,dyn} = \beta_0 + \beta_1 \cdot RW_{acc/sec} \quad (5)$$

where $RW_{acc/sec}$ represents the amount of accesses per second and β_0, β_1 are the regression coefficients.

We use both memory- and CPU-bounded SPEC CPU workloads, SPEC Power and *streamcluster* from PARSEC to test our model, using two of the lowest fan speeds (i.e. 1800RPM and 2400RPM). As these benchmarks do not stress the memory alone, the difference between model prediction and measured power, also reflects the power contribution of the other components of the server ($P_{other,dyn}$) besides the model error. All the test workloads result in RMSE below 10W, which is the error margin of the server power sensor. Therefore, our results have acceptable accuracy.

4.2 CPU temperature estimation

Using the previous models, we can obtain the server leakage power at a given temperature with sufficient accuracy. To adjust fan speed at runtime and minimize the energy consumption, we also need to predict the future temperature to compensate for the thermal delays associated with the processor. For this purpose, we propose a model which first predicts the steady-state temperature based on power measurements and fan speed, and then estimates the transient behavior.

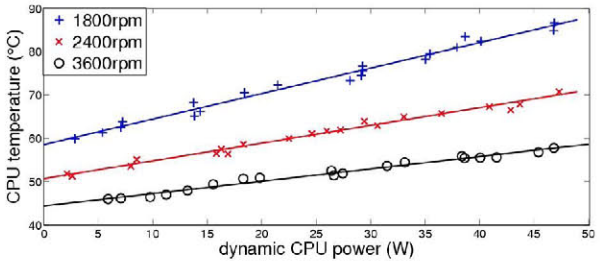

Fig. 8: Steady-state temperature model and measured samples for three different fan speeds.

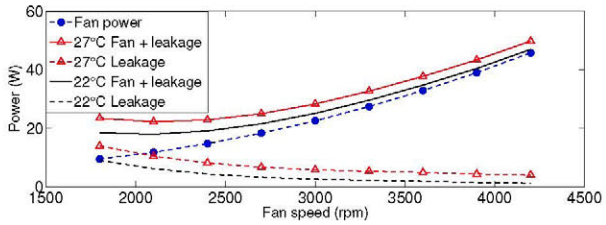

Fig. 9: Effect of ambient temperature on the *leakage power plus fan power* curve of mcf.

Steady-state estimation

The steady-state temperature of a processor running a constant workload is strongly correlated with dynamic power; i.e. each dynamic power level has a corresponding steady-state CPU temperature. To this end, we use our dynamic CPU power model derived in Section 4.1.

In our experiments, we observe a linear relationship between the steady-state maximum chip temperature and the dynamic power consumption for each fan speed as demonstrated in Figure 8. To train our model, we launch *LoadGen* with different duty cycles to vary the average dynamic power, and record the steady-state temperature. We repeat the procedure for each available fan speed and derive models in the following form:

$$T_{CPU,ss} = k_0 + k_1 \cdot P_{CPU,dyn} \quad (6)$$

where $T_{CPU,ss}$ is the steady-state CPU temperature, and k_0, k_1 are the model coefficients.

We derive our model using an ambient temperature of 22°C. However, as shown in Figure 9, ambient temperature affects the optimum fan speed, and including it in the model is necessary for robustness. To consider different ambient temperatures, we use the known linear relationship between the local ambient and the chip temperature [31]. We experimentally observe that if we add the difference in ambient temperature to our temperature estimation as an offset, the RMSE and maximum error do not increase. This approach ensures the robustness of the model while keeping its simplicity.

We validate our model by running a set of SPEC CPU2006 workloads at two different ambient temperatures, 22°C and 27°C, where we obtain a maximum error of 6.6°C and RMSE below 2.1°C. This accuracy is sufficient for our purposes.

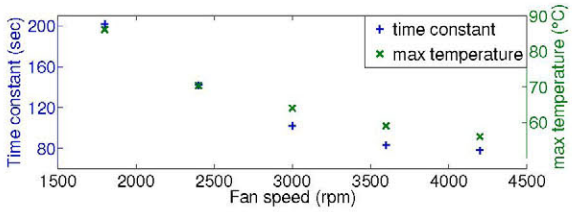

Fig. 10: Thermal time constant and maximum observed temperature under various fan speeds

Transient state modeling

When processor power varies, temperature changes exponentially with a time constant. We compute the thermal time constant of each fan speed by fitting exponential curves to the temperature measurements obtained while running *LoadGen* after the idle steady-state. As seen in Figure 10, the time constants, maximum observable temperatures and temperature range decrease as the fan speed increases. As the small changes in temperature do not affect the leakage power significantly, we only need to detect the large changes with time constants in the order of minutes. With such long time constants, we predict the temperature only during the next minute. A more fine-grained temperature prediction will lead to better approximations to the optimal fan speed by capturing small changes in the temperature; however, it will also induce unnecessary changes in fan speed and decrease the lifetime of the fans. The duration of the temperature prediction should be selected considering this trade-off.

4.3 Models Summary

To summarize the work presented in this section:

- We have modeled the contributors to power consumption that are affected by leakage power and workload allocation, i.e. CPU and memory power.
- We have estimated the steady-state and the transient CPU temperature.

Given a certain workload, the models allow us (i) to separate the contribution of dynamic power from that of leakage, (ii) to predict CPU temperature and thus leakage power for each available fan speed, and (iii) to select the fan speed that minimizes the *leakage plus fan* power. Moreover, the models enables us to evaluate the impact of workload allocation in Section 6.

5 Fan Control Policy

Our fan control policy uses temperature and power measurements to proactively determine the fan speed that minimizes the *fan plus leakage* power. In this section, we describe our policy, its overhead and applicability.

Figure 11 shows the fan speed selection procedure for steady-state. As discussed in Section 4.1, the dynamic power of a constant workload can be estimated by subtracting the temperature-dependent leakage and

idle power from the CPU power. Using dynamic power, we predict the steady-state processor temperature under each available fan speed setting using our temperature model given in Section 4.2. Then, we calculate the expected steady-state leakage and fan power for every fan speed. Finally, we set the fan speed to the value that provides the minimum *leakage plus fan* power.

The workloads that we use present small fluctuations in power consumption that do not affect temperature significantly. To avoid inaccuracies caused by these fluctuations, we average dynamic power over a period significantly smaller than the thermal time constants. In our case, we choose an 8-second averaging that captures the large changes in power that govern temperature while smoothing out the power trace.

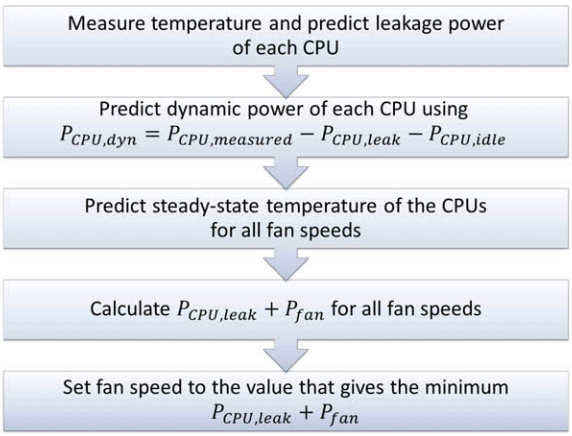

Fig. 11: Fan speed selection procedure for steady-state.

The main shortcoming of a steady-state approach is that it ignores the thermal transients. As the thermal time constants are large, the processor temperature can differ from its steady-state value by several degrees, especially for highly variable workloads, and energy can be saved during transients. We consider the transient behavior by proposing the run-time policy given in Algorithm 1, where f represents a function and f^{-1} its inverse.

The policy first calculates the expected steady-state temperature T_{ss} for each CPU (lines 2-3). Then, it computes the average temperature T_{pred} over the next τ_{wait} period, using a closed form integration of the transient temperature prediction (line 4). As temperature changes slowly, we find a dynamic power corresponding to T_{pred} using our steady-state temperature model inversely, obtaining dynamic power (line 5).

We use dynamic power $P_{dyn,pred}$ to predict the expected temperature T_{exp} under each fan speed (line 7). Next, the expected leakage power $P_{leakT,exp}$ under various fan speeds are calculated using the leakage model. We prevent the selection of fan speeds that result in temperatures above the critical value $T_{critical}$, by setting the corresponding leakage power to a very high value.

All predictions until this point are done separately for each CPU. Finally, total server leakage plus fan power consumption $P_{leakT+fan}$ is computed for all available fan speeds. The policy selects the fan speed that provides the minimum $P_{leakT+fan}$ and waits τ_{wait} seconds while monitoring the system for a workload change. If the dynamic CPU power changes significantly, this interval is interrupted and the optimum fan speed is re-calculated. This waiting time ensures the stability of the controller and prevents the fan reliability issues that could arise with very frequent fan speed changes (i.e. in the order of seconds). For our system, we choose a τ_{wait} value of 1 minute, which is a safe choice considering the large thermal time constants.

We use a 300RPM resolution for the fan speed selection in our policy, which is a heuristically selected value. This resolution is selected such that the available fan speeds lead to a sufficient approximation to the optimal cooling conditions. Selecting an unnecessarily fine resolution will increase the computational overhead of the policy.

Algorithm 1 Policy

1: **for** each CPU p **do**
2: $P^p_{dyn} = P^p_{meas} - P^p_{idle} - f_{leakage\_model}(T^p_{meas})$
3: $T^p_{ss} = f_{temperature\_model}(P^p_{dyn})$
4: $T^p_{pred} = f_{transient\_model}(T^p_{ss}, T^p_{meas})$
5: $P^p_{dyn,pred} = f^{-1}_{temperature\_model}(T^p_{pred})$
6: **for** each fan speed s **do**
7: $T^{p,s}_{exp} = f_{temperature\_model}(P^p_{dyn,pred})$
8: **if** $T^{p,s}_{exp} \geq T_{critical}$ **then**
9: $P^{p,s}_{leakT,exp} = \infty$
10: **else**
11: $P^{p,s}_{leakT,exp} = f_{leakage\_model}(T^{p,s}_{exp})$
12: **end if**
13: **end for**
14: **end for**
15: **for** each fan speed s **do**
16: $P^s_{leakT+fan} = P^s_{fan} + \sum_p P^{p,s}_{leakT,exp}$
17: **end for**
18: Set fan speed to $\underset{s}{\text{argmin}}(P^s_{leakT+fan})$
19: Wait τ_{wait} while monitoring P_{dyn}

Applicability

Our models and the fan speed algorithm are based solely on power and temperature measurements. Thus, they can be derived using any constant stress workload with controllable utilization. Even if sensor data are limited by a measurement delay, the performance of our policy is not significantly affected as the thermal time constants are in the order of minutes, which is much slower than the policy response time. Even though our policy does not consider hot spots on the chip, the operating points are well below the critical thresholds.

Overhead

The fan speed control policy is run by the DLC-PC in our implementation. On the DLC-PC, the policy measures

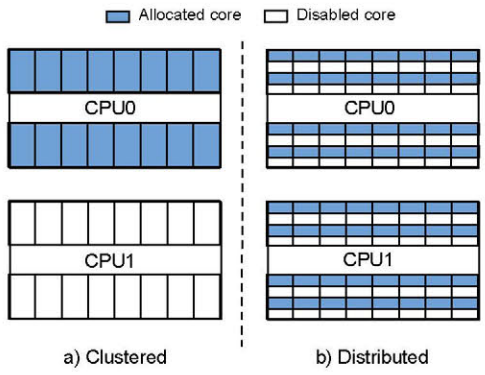

Fig. 12: Clustered vs distributed allocation schemes for 128 active threads.

and averages power every second, and decides on the fan speed every 60 seconds using LUTs and polynomials. The leakage and temperature prediction is computed only for 9 different fan speeds that cover the entire fan speed range (from 1800 to 4200RPM) with a resolution of 300RPM. As these are very simple operations with long periods, the policy has negligible overhead and can be easily implemented in the service processor.

6 Impact of Workload Allocation

This section describes the impact of workload allocation on the leakage-cooling and energy-performance tradeoffs at the server level.

6.1 Allocation schemes

We experiment with two different allocation schemes: *clustered* and *distributed*. Clustered allocation packs all the threads together into the first N cores of the server, maximally utilizing all the available hardware threads in a core. Distributed allocation spreads the workload as much as possible into all available cores. Figure 12 shows a diagram of the clustered and distributed allocation for an application with 128 threads, i.e., 50% utilization. Each box in the figure represents a core, each with 8 hardware threads that can be individually enabled or disabled. In order to get different utilization values in our server, we launch multiple copies of the benchmarks. For example, 64 copies of a single-threaded SPEC CPU benchmark utilize 25% of the available hardware threads.

Distributing the workload activates more cores and increases the number of available FP units and integer pipelines, as well as the amount of cache and memory bandwidth. On the other hand, clustering the workload reduces the amount of active cores in the server, decreasing the power consumption. Recent enterprise servers come with elastic power policies and core-disabling capabilities that allow to set idle cores in a deep sleep mode when all their hardware threads are disabled. For similar architectures, when all threads in a CPU are idle, up to 60% of power can be saved by setting cores in a deep sleep state [23].

Task	Allocation	$P_{CPU,dyn}$ (W)	P_{mem} (W)	T_{CPU0} ($2400rpm, ^\circ C$)	T_{CPU1} ($2400rpm, ^\circ C$)
sjeng	Distributed	55.2	32	57.4	55.4
192 threads	Clustered	47.4	31	59.4	52.9
mcf	Distributed	14.9	95	54.3	51.5
192	Clustered	14.2	98	56.5	51.4
calculix	Distributed	75.1	114	63.4	60.7
192	Clustered	65.1	99	66.4	55.6
bodytrack	Distributed	30.0	16	55.2	53.1
192	Clustered	26.3	18	54.7	50.6

TABLE 1: Summary of dynamic power and CPU temperature at 2400RPM for selected PARSEC and SPEC benchmarks running with 192 threads

6.2 Leakage-cooling tradeoffs

From the leakage-cooling perspective, distributed allocation reduces the CPU temperature by balancing the workload across all cores. This generates similar temperatures in both CPUs, and thus, similar leakage power. However, clustering the workload stresses CPU0 more than CPU1, and generates temperature and leakage imbalance between the CPUs. Thus, the same workload can yield different optimum fan speed values depending on the allocation scheme. Table 1 shows the impact of allocation for four workloads with different characteristics from our test set with 75% utilization, all running under the same fan speed. As can be seen, the temperature imbalance between the CPUs depends on the workload, and leads to different optimum fan speeds. For example, the optimum fan speed for *Bodytrack* is 1800RPM when the clustered allocation is selected, but 2400RPM if we select the distributed scheme.

As workload allocation changes the leakage and cooling tradeoffs, fan speed policies that do not take into account temperature and power imbalances cannot fully exploit the advantage of energy efficient dynamic fan control. Our proactive policy, on the contrary, is robust to workload imbalances across the CPUs as it predicts the leakage for both CPUs separately and computes the optimum fan speed. Therefore, the policy finds the optimum regardless of how the workload is allocated.

6.3 Energy-performance tradeoffs

Interesting tradeoffs exist in terms of performance and energy when clustering/distributing workloads. Distributed allocation leads to a flatter thermal profile and maximally utilizes pipelines, whereas clustering reduces the communication distance between threads, and may increase the performance of parallel applications with data sharing.

We use the Energy-Delay Product (EDP) metric, which weighs power against the square of execution time, for the joint evaluation of performance and energy. EDP is calculated considering the total CPU power (P_{CPU}) and memory power ($P_{mem,dyn}$), as those are the two main factors affected by the workload allocation. We assume that when all the hardware threads in a core are disabled, the core goes into deep sleep mode and its idle power $P_{CPU,idle}$ is reduced.

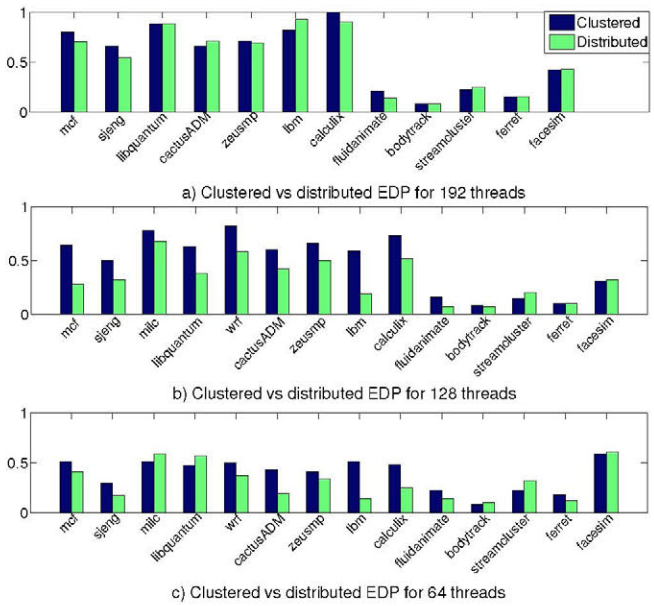

Fig. 13: Normalized EDP in clustered and distributed allocation schemes for SPEC CPU and PARSEC benchmarks under various number of threads.

Type	Workload	Perf. counters for distributed			
		IPC	Mem Acc.	FP Instr	L1 acc
High IPC	sjeng	1.0	0.01	0.0	0.3
Mem. intensive	mcf	0.1	0.9	0.0	0.8
High FP Instr.	calculix	0.7	0.1	0.6	0.7
Low L1 & L2 misses	bodytrack	0.3	0.0	0.4	0.1

TABLE 2: Summary of performance counters (normalized to the highest value across benchmarks) of selected PARSEC and SPEC benchmarks with 192 threads.

Figure 13 presents the EDP comparison between the two allocation schemes for the benchmarks in our test set under various utilization values. The plot is normalized to the highest EDP value across experiments. We see that distributing is better for most cases, as the workloads benefit from a larger number of available computational units. As expected, results for clustering and distributing converge as the number of threads increase. Note that for the cases where EDP are similar (e.g., *libquantum* with 192 threads), leakage-cooling tradeoffs should be considered when determining the most efficient allocation.

Because the results are highly dependent on the workload characteristics, we gather relevant performance counters to explain the differences in EDP between the two allocation schemes. First, we perform a correlation test over the performance counters when running the distributed allocation scheme. We find the following metrics to have high correlation with EDP: *active thread count, IPC, L1 cache misses, FP instructions, store instructions and memory accesses*. All metrics except *active thread count* and *IPC* are computed per instruction, and normalized to the highest observed value. Table 2 summarizes the most relevant features for some workloads. Table 3 groups together the benchmarks that exhibit similar characteristics in terms of their performance counters, and thus, exhibit similar tradeoffs in EDP.

Metric	Benchmark
High IPC	sjeng, fluidanimate, calculix, ferret
High FP Instr.	lbm, zeusmp, cactusADM, calculix, wrf
Memory intensive	lbm, mcf, libquantum, milc
Low L1 & L2 misses	bodytrack, fluidanimate, canneal

TABLE 3: Summary of relevant characteristics for SPEC and PARSEC benchmarks. For each parameter, benchmarks are ordered from high-to-low.

Putting together the experimental results of Figure 13, Table 2, and Table 3, we see that high-IPC CPU-bounded workloads such as *sjeng*, *fluidanimate* or *calculix* always benefit more from distributing, regardless of utilization. *zeusmp* and *cactusADM* do not have the highest IPC values, but they are FP-intensive. Because each core shares one FP unit among all threads, as utilization decreases, these benchmarks benefit more from being distributed. Benchmarks such as *streamcluster* have a high amount of synchronization locks between threads, so they do not benefit from a higher number of available pipelines, and are better clustered for all utilization cases. The performance of memory-intensive applications, such as *mcf, lbm, libquantum, milc* depends on the tradeoff between available memory bandwidth (decreases with higher utilization) and contention (increases with higher utilization).

In a more general way, we can highlight the following results regarding task classification according to EDP:

- High-IPC and high-FP non-memory-intensive workloads achieve lower EDP when they are distributed for all utilization values. However, benefits are higher as utilization decreases.
- Low-IPC non-memory-intensive tasks (i.e., tasks with many synchronization locks) have lower EDP when they are clustered for all utilization levels.
- Memory-intensive benchmarks benefit more from distributing, especially for medium utilization values, because of the tradeoffs between available memory bandwidth and contention.

7 Results

In this section, we present several state-of-the-art policies and compare their performance against our proposed proactive fan control strategy. We also show the tradeoffs in terms of energy and performance when using different allocation schemes and how our proactive fan control policy is robust to power and temperature imbalances. Moreover, by using the power consumption traces of a real data center we provide an estimation of the expected benefits of our solution at the data center scope.

7.1 Baseline policies

Best fixed fan speed

The default server fan policy sets a fixed fan speed that ensures the server reliability for a worst-case scenario for each ambient temperature. The default fan speed for our server is of 3000RPM, which leads to significant overcooling when the ambient temperature is low.

To have a fair comparison, instead of 3000RPM, we use as a baseline the fan speed that minimizes *leakage plus fan* power for the majority of the workloads to evaluate the benefits of dynamic fan speed selection. After running all workloads under all fan speeds, we find that the best fixed fan speed in our system is 2400RPM for 22°C ambient temperature.

TAPO

The TAPO server fan control policy introduced by Huang et al. [13] changes the thermal set point T_{sp} of the processor to indirectly control the fan speed. Assuming the workload is constant, once the thermal steady-state is reached, the policy changes T_{sp}. Then, it observes the change in the processor temperature and power processor to decide whether to increase or decrease the setpoint to achieve lower power.

TAPO assumes an underlying fan controller that keeps the maximum processor temperature at T_{sp}. In our TAPO implementation, we write a bang-bang fan speed control script that checks CPU temperature every Δt minutes and changes the fan speed if the temperature is out of the range $T_{sp} \pm \Delta T$. ΔT and Δt are heuristically chosen as 5°C and 2 minutes, respectively, to avoid fan speed oscillations. The fan speed resolution is 300RPM as in our proactive policy.

Bang-bang controller

The bang-bang controller tracks CPU temperature and tries to maintain the temperature within a desirable range by means of a multi-threshold controller. Our implementation tries to keep temperature within the $65°C$-$75°C$, thus: (i) if maximum temperature T_{max} goes below $60°C$, fan speed is set to 1800RPM (lowest); (ii) if T_{max} is in between $60°C$ to 65 $°C$, fan speed is lowered by 600RPM; (iii) if T_{max} is between 65 to 75 degrees, no action is taken; (iv) if T_{max} rises above $75°C$, fan speed is increased by 600RPM; and, (v) if T_{max} is above $80°C$, fan speed is increased to 4200RPM.

The threshold values are heuristically chosen to optimize the tradeoff between high fan speed change frequency and high temperature overshoots [14], ensuring the stability of the controller while keeping temperature in a range that ensures high reliability and low leakage.

LUT-based fan control

This policy implements the idea presented in our previous work [14]. The controller monitors load periodically and tries to minimize the leakage plus cooling power by setting the optimum fan speed during run-time depending on the utilization of the server. For that purpose, a LUT that holds the optimum fan speed values for each utilization value is generated using *LoadGen*.

Because the controller makes decisions based on changes in the load utilization rather than reacting to temperature changes, the system proactively sets fan speed before a thermal event occurs. To ensure the stability of the controller and to prevent fan reliability issues in the case of highly variable workloads, we do not allow high-to-low RPM changes for 1 minute after each RPM update.

7.2 Workload profiles

We generate 4 different workload profiles that exhibit a wide range of behaviors from a statistical perspective to evaluate our method against existing policies. Each workload profile consists of 10 tasks of our test set (i.e., the workloads from SPEC or PARSEC launched with certain number of copies as described in Sections 4 and 6), generated with a Poisson statistical distribution of arrival (λ) and service (μ) times. To generate profiles with variable stress in terms of power consumption, all benchmarks from SPEC and PARSEC with 25%, 50% and 75% utilization are arranged into two classes: high power consumption and low power consumption. For each profile, we vary the probability of choosing benchmarks from the high power class ($p(high)$). Within each class, benchmarks are chosen randomly.

Table 4 summarizes the main parameters of each profile, and describes the sequence of benchmarks.

7.3 Joint workload and cooling management

We implement the fan control policies described in Section 7.1 plus our proposed policy, and test them for every workload profile described in Section 7.2, under different allocation policies: (i) a clustered allocation scheme without core sleep states, (ii) a clustered allocation scheme with core sleep states, (iii) a distributed allocation scheme, and (iv) a best-case allocation that selects the lowest EDP allocation for each benchmark, as in Figure 13.

Table 5 shows the results of all the controllers for the clustered (without core sleep states) and the distributed allocation schemes. The energy metric (column 4) is computed with total CPU power minus CPU idle power plus fan power (i.e., $P_{CPU+fan} = P_{CPU} - P_{CPU,idle} + P_{fan}$), and the savings (column 5) represent the % reduction of leakage and fan energy achieved by our policy compared to other policies. It has to be taken into account that the fixed fan speed policy shown in Table 5 has been selected considering the leakage-cooling tradeoffs (see Section 7.1). This policy is already reducing the CPU energy of workload profile 1 by 8.3% when compared to the server default fan control policy.

The performance of the fan control policies depend both on the workload profile and on the allocation. As the fixed fan speed policy uses 2400RPM, its performance mainly depends on the number of the benchmark-allocation pairs, that have 2400RPM as their best fan

Profile	Arrival λ (min), Service μ (min)	Workload sequence (benchmark and number of threads) 1	2	3	4	5	6	7	8	9	10
1	25 , 20	ferret 128	libquan. 192	zeusmp 128	wrf 128	calculix 128	fluid. 192	sjeng 128	cactus 128	facesim 128	zeusmp 128
2	25 , 20	zeusmp 192	milc 128	wrf 128	sjeng 128	mcf 128	cactus 128	calculix 64	lbm 128	canneal 64	zeusmp 192
3	15 , 10	sjeng 192	mcf 128	sjeng 128	sjeng 128	calculix 192	facesim 192	facesim 128	facesim 128	ferret 128	facesim 128
4	15 , 10	fluid. 192	canneal 128	stream. 64	stream. 64	canneal 128	calculix 128	canneal 64	canneal 64	lbm 192	bodytrack 192

TABLE 4: Summary of main characteristics for workload profiles. The profiles 1 and 3 have a $p(high)$ of 0.8, and the profiles 2 and 4 have a $p(high)$ of 0.2.

speed. For example, the fixed fan policy performs better with the clustered allocation than the distributed allocation while running workload profile 3, because most of the applications in profile 3 have smaller energy consumption when clustered. The fixed fan speed policy outperforms the dynamic baseline policies in some cases, as temperature-driven controllers (i.e., TAPO and Bang-Bang) use the maximum temperature across two CPUs to set the fan speed. As they do not consider the temperature imbalance between CPUs, their performance depends on how well the total leakage is described by the maximum temperature. On the other hand, the LUT controller uses utilization to set the fan speed. As utilization is not an accurate metric for power modeling, it does not perform well with arbitrary workloads. Our proactive policy computes the fan speed that minimizes the sum of the cooling power and the leakage power of both CPUs, and thus, it yields the most efficient results regardless of the workload and the allocation.

Figure 14 shows the fan speed and the processor temperature trends of the fixed fan speed policy, the bang-bang policy, and the proactive policy running workload profile 1 under clustered allocation scheme. We observe that the proactive policy reduces oscillations in temperature when compared to the bang-bang controller, as it maintains temperature within the range that minimizes the leakage plus fan power curve.

Finally, we compare the energy consumed by the workload profiles under different allocation schemes. Even though the SPARC T3 cores support core-level deep sleep modes, the current software on our server does not support direct control over this feature. To overcome this limitation, we use the reported sleep power values [23], and compute EDP for the scenarios including sleep accordingly. We apply this computation adjustment to the real data obtained on our system.

Table 6 shows a summary of EDP, energy, power and performance metrics for different allocation policies under the proactive fan control policy. The energy results for columns 3 and 4 are computed by summing up memory power and total CPU power. Column 5 reports workload execution time without considering the idle server time between workload arrivals. The best-case allocation shows the lowest energy consumption in most of the cases, resulting in up to 12.7% improvement when compared to a distributed allocation and up to 15% when compared to a clustered allocation scheme. Even though the execution time of the best-case allocation is longer than the distributed scheme in all cases, it results in better EDP by saving more energy. Moreover, the best-case allocation reduces the maximum CPU temperature when compared to the clustered allocation, and increases only by a maximum of $1°C$ when compared to the distributed allocation.

Profile, Allocation	Fan policy	Fan Energy (Wh)	CPU Energy (Wh)	Leak+Fan Savings (%)	Avg RPM
1, Clustered	Fixed	64.2	243.9	2.3	2400
	TAPO	42.4	243.8	2.2	1848
	Bang	44.1	241.2	0.2	1888
	LUT	43.9	245.4	3.4	1883
	Proactive	49.8	240.9	-	2047
1, Distributed	Fixed	64.2	241.2	1.3	2400
	TAPO	41.6	241.5	1.6	1821
	Bang	41.4	243.0	2.7	1819
	LUT	43.9	243.6	3.2	1883
	Proactive	57.3	239.5	-	2236
2, Clustered	Fixed	62.3	217.7	3.1	2400
	TAPO	42.1	216.2	1.9	1874
	Bang	43.8	216.4	2.0	1915
	LUT	44.1	217.6	3.0	1921
	Proactive	55.3	213.9	-	2226
2, Distributed	Fixed	62.3	219.4	2.5	2400
	TAPO	40.2	219.6	2.6	1821
	Bang	40.4	218.1	1.5	1825
	LUT	43.5	219.4	2.5	1906
	Proactive	54.6	216.3	-	2210
3, Clustered	Fixed	33.1	137.8	0.8	2400
	TAPO	22.7	137.7	0.6	1887
	Bang	22.9	138.3	1.5	1896
	LUT	26.6	138.1	1.2	2079
	Proactive	30.9	137.3	-	2297
3, Distributed	Fixed	33.1	143.0	6.4	2400
	TAPO	22.7	140.1	2.4	1887
	Bang	22.4	140.5	3.0	1872
	LUT	25.8	139.6	1.7	2039
	Proactive	28.2	138.5	-	2171
4, Clustered	Fixed	58.5	163.1	2.7	2400
	TAPO	38.2	161.7	1.5	1843
	Bang	38.0	161.9	1.7	1837
	LUT	41.8	162.0	1.7	1927
	Proactive	47.9	160.1	-	2120
4, Distributed	Fixed	58.0	164.8	3.7	2400
	TAPO	37.7	162.5	1.6	1830
	Bang	38.0	162.3	1.5	1837
	LUT	36.9	163.4	2.5	1806
	Proactive	47.8	160.7	-	2118

TABLE 5: Summary of fan control results for all workloads under different allocation schemes.

7.4 Discussion on the Impact at the Data Center

Finally, we discuss and evaluate the impact of our server-level policies at the data center scope. To this end, we gather server power traces of a high-performance

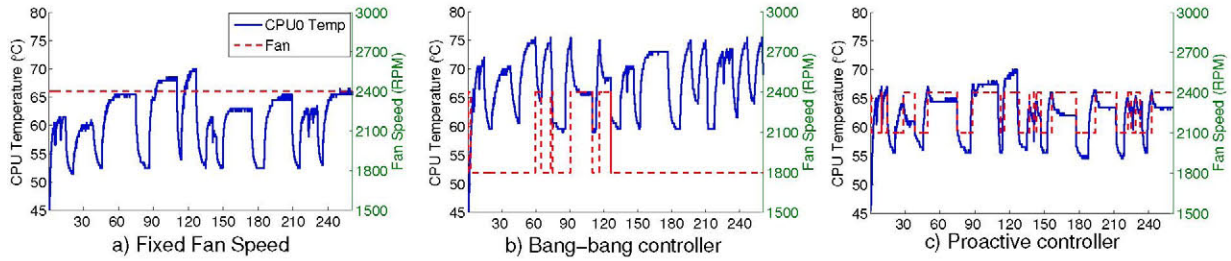

Fig. 14: Fixed speed, bang-bang, and proactive controller temperature and RPM traces for workload profile 1.

Profile, Allocation	EDP (kWh^2)	Energy (Wh)	Exec.Time (min)	$T_{CPU_{max}}$ (°C)
1, Clustered (w/o sleep)	2.63	823.8	192	71
Clustered (w/sleep)	2.33	731.6	192	71
Distributed	1.83	712.4	154	67
Best-case	1.82	697.6	157	67.5
2, Clustered (w/o sleep)	2.87	818.0	210	68
Clustered (w/sleep)	2.48	707.9	210	68
Distributed	1.84	697.6	158	66
Best-case	1.86	684.5	163	65
3, Clustered (w/o sleep)	0.74	428.1	104	71.5
Clustered (w/sleep)	0.67	383.7	104	71.5
Distributed	0.55	384.4	85	68.5
Best-case	0.55	368.5	89	67.5
4, Clustered (w/o sleep)	2.1	634.3	196	71
Clustered (w/sleep)	1.7	525.7	196	71
Distributed	1.9	617.5	182	64
Best-case	1.7	538.8	191	64.5

TABLE 6: EDP, Energy and performance for various allocation policies with proactive policy.

computing cluster consisting of 260 computer nodes in 9 racks at the Madrid Supercomputing and Visualization Center (CeSViMa). By using the data center telemetry deployed in CeSViMa, we gather 3 hours of real server power traces for 256 servers. We use this power traces to simulate the execution of our proactive policy in a larger-scale scenario with a realistic workload profile.

We compute the savings that our policy would achieve when compared to the fixed fan speed policy (see Section 7.1). We do so for the whole cluster under different room ambient temperatures that are within the allowable range published by ASHRAE (i.e., 5°C to 45°C for class A4 volume servers, and 15°C to 32°C for class A1 enterprise servers). We perform this analysis off-line in simulation space, applying the models and policies to the gathered power traces and computing the energy savings. For every single power trace and ambient temperature, our policy outperforms the fixed fan speed policy and the default server fan policy. For 22°C ambient temperature, our policy obtains 1.9% energy savings for the whole cluster in *leakage plus fan* power when compared to a fixed fan speed of 2400RPM. If room temperature increases to 27°C, our results show that a fixed fan speed of 2400RPM is no longer valid and needs to be raised to 2700RPM to ensure the worst-case server temperature. Thus, energy savings for the proactive policy increase to 5.5% when compared to the fixed fan speed policy. Similarly, if we further increase ambient temperature to 32°C, the fan speed for the fixed fan control policy needs to be raised to 3000RPM, and the savings for our proactive fan control policy achieve 10.3%. These savings are translated into a reduction of 2.5% in the total CPU energy consumption of the cluster for an ambient temperature of 27°C.

As room temperature raises, the fan speed needed to keep servers within safe environmental conditions also increases. Moreover, the sum of leakage and fan power increases by 20% when room temperature raises from 22°C to 32°C for the proactive fan control policy. This observation is in accordance with prior work [9]. Moreover, the impact of the leakage-temperature tradeoffs in the overall power consumption of the data center increases as the data room cooling becomes more efficient. In data centers with a high PUE value (i.e., around 2), increasing the room temperature yields important energy savings. However, in data centers with improved cooling subsystems (based on hot/cold aisle containment or free cooling), raising room ambient temperature yields less savings because of the increase in the *leakage plus fan* power. Figure 15 shows the total energy consumption in CeSViMa, assuming different PUE scenarios and assuming that each degree of increase in room temperature yields 4% energy savings in the cooling subsystem, which is a commonly accepted metric in industry [32]. As we can see, if the initial PUE is 2.0 for a room at 22°C ambient temperature, the reduction in energy consumption when increasing room temperature is very significant. For the world average PUE of 1.65, increasing room temperature still yields significant savings. However, for lower PUE values (achieved by improving cooling efficiency), increasing room temperature leads to reduced energy savings, because of the impact of leakage and fans in the overall energy consumption of the data center. Thus, as data room cooling becomes more efficient, the impact of the leakage-cooling tradeoffs at the server level becomes more significant.

8 Conclusions

Improving server cooling efficiency is a necessary prerequisite for sustainable computing at data centers, which have prohibitively high electricity use. Higher chip power densities brought by new process technologies cause temperatures to rise, which in turn, increases leakage power. Moreover, as data center cooling becomes more efficient, the contribution of leakage and server fans

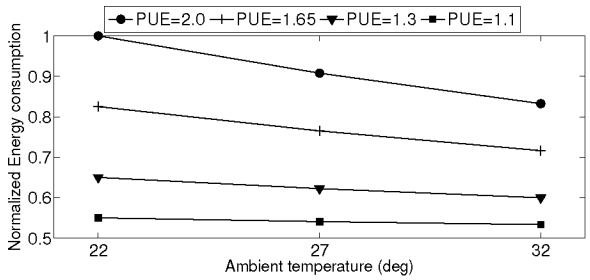

Fig. 15: Normalized CeSViMa cooling plus IT power consumption for the workload execution under various PUE scenarios

become more significant. These observations imply that the tradeoffs between cooling and leakage power need to be taken into account for designing efficient cooling strategies both at server and at data center levels.

In this paper, we have developed power models that accurately estimate various contributors to server power consumption. Using these models, we have proposed a leakage-aware cooling control policy that minimizes the energy consumption. Our policy is able to work with arbitrary workloads, and it is robust to variations in workload allocation. Our results on a commercial server show that our policy reduces the *leakage plus fan* energy by up to a 6% compared to existing policies and the CPU energy consumption by more than 9% compared to the default server control policy, without imposing any performance penalty. We have also analyzed the impact of workload allocation, and have shown that by choosing the best-EDP allocation for a given load along with using our proactive cooling control policy, we can obtain energy savings by up to 15%.

The devised policy has also been applied in a broader distributed scenario with real data center traces, optimizing CPU power consumption by 2.5% for the whole cluster, and showing how the impact of our policy raises as data room temperature increases.

ACKNOWLEDGMENTS

Research by Marina Zapater has been partly supported by a PICATA predoctoral fellowship of the Moncloa Campus of International Excellence (UCM-UPM). This work has been partly funded by the Spanish Ministry of Economy and Competitivity under research grant TEC2012-33892, by Oracle Corp., and Decision Detective Corporation (SBIR). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa).

REFERENCES

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010," Analytics Press, Oakland, CA, Tech. Rep., 2011.

[2] A. V. ComputerWeekly.com, "Global census shows datacentre power demand grew 63% in 2012," http://www.computerweekly.com/news/2240164589/Datacentre-power-demand-grew-63-in-2012-Global-datacentre-census, October 2012.

[3] T. Breen, E. Walsh, J. Punch, A. Shah, and C. Bash, "From chip to cooling tower data center modeling: Part I influence of server inlet temperature and temperature rise across cabinet," in *ITherm*, 2010, pp. 1–10.

[4] J. K. Matt Stansberry, "Uptime institute 2013 data center industry survey," Uptime Institute, Tech. Rep., 2013.

[5] A. P. M. V. Antonio Celesti, Francesco Tusa, "Towards energy sustainability in federated and interoperable clouds," in *Sustainable Practices: Concepts, Methodologies, Tools and Applications*, March 2014, pp. 279–301.

[6] G. Inc., "Efficiency: How we do it." [Online]. Available: http://www.google.com/about/datacenters/efficiency/internal/

[7] M. Iyengar and R. Schmidt, "Analytical modeling for thermodynamic characterization of data center cooling systems," *Journal of Electronic Packaging*, vol. 113, Feb. 2009.

[8] Y. Xie and W. Hung, "Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (MPSoC) design," *The Journal of VLSI Signal Processing*, vol. 45, no. 3, pp. 177–189, 2006.

[9] M. Patterson, "The effect of data center temperature on energy efficiency," in *Thermal and Thermomechanical Phenomena in Electronic Systems, (ITHERM)*, May 2008, pp. 1167 –1174.

[10] A. Coskun, T. Rosing, and K. Gross, "Utilizing predictors for efficient thermal management in multiprocessor socs," *TCAD*, vol. 28, no. 10, pp. 1503 –1516, 2009.

[11] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: adaptive dvfs and thread packing under power caps," in *MICRO*, 2011, pp. 175–185.

[12] X. Han and Y. Joshi, "Energy reduction in server cooling via real time thermal control," in *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)*, March 2012, pp. 20 –27.

[13] W. Huang, M. Allen-Ware, J. Carter, E. Elnozahy, H. Hamann, T. Keller, C. Lefurgy, J. Li, K. Rajamani, and J. Rubio, "TAPO: Thermal-aware power optimization techniques for servers and data centers," in *IGCC*, 2011, pp. 1–8.

[14] M. Zapater, J. L. Ayala, J. M. Moya, K. Vaidyanathan, K. Gross, and A. K. Coskun, "Leakage and temperature aware server control for improving energy efficiency in data centers," in *DATE*, 2013.

[15] D. Shin, J. Kim, N. Chang, J. Choi, S. W. Chung, and E.-Y. Chung, "Energy-optimal dynamic thermal management for green computing," in *ICCAD*, 2009, pp. 652–657.

[16] C. S. Chan, Y. Jin, Y.-K. Wu, K. Gross, K. Vaidyanathan, and T. Rosing, "Fan-speed-aware scheduling of data intensive jobs," in *ISLPED*, 2012, pp. 409–414.

[17] B. Pradelle, N. Triquenaux, J. Beyler, and W. Jalby, "Energy-centric dynamic fan control," *Computer Science - Research and Development*, pp. 1–9, 2013.

[18] A. Lewis and et al., "Run-time energy consumption estimation based on workload in server systems," in *HotPower*, 2008, pp. 4–4.

[19] X. Fan *et al.*, "Power provisioning for a warehouse-sized computer," in *ISCA*, 2007, pp. 13–23.

[20] A. Bohra and V. Chaudhary, "VMeter: Power modelling for virtualized clouds," in *IPDPSW*, 2010, pp. 1–8.

[21] R. Cochran, C. Hankendi, A. Coskun, and S. Reda, "Identifying the optimal energy-efficient operating points of parallel workloads," in *ICCAD*, 2011, pp. 608–615.

[22] R. Ayoub, R. Nath, and T. Rosing, "JETC: joint energy thermal and cooling management for memory and cpu subsystems in servers," in *HPCA*, feb. 2012, pp. 1 –12.

[23] J. Shin *et al.*, "A 40nm 16-core 128-thread cmt sparc soc processor," in *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2010, pp. 98 –99.

[24] K. Gross, K. Whisnant, and A. Urmanov, "Electronic prognostics through continuous system telemetry," in *MFPT*, April 2006, pp. 53–62.

[25] R. Longbottom, "Randmem memory benchmark," http://www.roylongbottom.org.uk/, October 2012.

[26] SPEC Power Committee, "Spec power benchmark 2008," http://www.spec.org/power_ssj2008/, July 2012.

[27] SPEC CPU Subcommittee and John L. Henning, "SPEC CPU 2006 benchmark descriptions," http://www.spec.org/cpu2006/.

[28] A. Phansalkar, A. Joshi, and L. K. John, "Subsetting the spec cpu2006 benchmark suite." *SIGARCH Computer Architecture News*, vol. 35, no. 1, pp. 69–76, 2007.

[29] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in *PACT*, 2008, pp. 72–81.

[30] R. Ayoub, K. Indukuri, and T. Rosing, "Temperature aware dynamic workload scheduling in multisocket cpu servers," *TCAD*, vol. 30, no. 9, pp. 1359–1372, 2011.

[31] M. Pedram and S. Nazarian, "Thermal modeling, analysis, and management in VLSI circuits: Principles and methods," *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1487–1501, 2006.

[32] R. Miller, "Data center cooling set points debated," http://www.datacenterknowledge.com/archives/2007/09/24/data-center-cooling-set-points-debated/, September 2007.

Marina Zapater is currently a PhD Candidate in the Electronic Engineering Department at the Universidad Politécnica de Madrid, Spain. She has been awarded a Research Grant by the Program for Attracting Talent (PICATA) of the Campus of International Excellence of Moncloa. She received a MSc in Telecommunication Engineering and a MSc in Electronic Engineering from the Universitat Politècnica de Catalunya, Spain, in 2010. Her current research focuses on proactive and reactive thermal optimization of complex heterogeneous systems, and energy efficiency in data centers.

Ozan Tuncer is currently a PhD candidate in the Department of Electrical and Computer Engineering, Boston University, Boston, MA. He received his B.S. degree in Electrical and Electronics Engineering from the Middle East Technical University, Ankara, Turkey. His current research interests are thermal management and energy efficiency in data centers and multicore systems.

Jose L. Ayala is currently an Associate Professor in the Department of Computer Architecture and Automation at the Complutense University of Madrid. He received his MSc and PhD degrees in Telecommunication Engineering from the Technical University of Madrid, Spain, in 2001 and 2005, respectively. He is member of the HiPEAC European Network of Excellence, IEEE and ACM. He has organized several international events as General Chair and Program Chair, such as VLSI-SoC, GLSVLSI and PATMOS. He has served as TPC member of many conferences, including DATE, DAC, VLSI-SoC, ICCAD, etc. He has lead a large number of international research projects and bilateral projects with industry, in the fields of power and energy optimization of embedded systems, and non-invasive health monitoring. His current research interests are thermal- and energy-aware design and management of processor-based systems, design of embedded processors, thermal estimation, 3D integration, health monitoring and wireless sensor networks.

Jose M. Moya is currently an Associate Professor in the Department of Electronic Engineering, Technical University of Madrid. He received his MSc and PhD degrees in Telecommunication Engineering from the Technical University of Madrid, Spain, in 1999 and 2003, respectively. He is member of the HiPEAC European Network of Excellence and the Spanish technological platforms eSEC and PROMETEO. He has served as TPC member of many conferences, including DATE, IEEE ICC, IEEE MASS, IEEE GLOBECOM, DCOSS, etc. He has participated in a large number of national research projects and bilateral projects with industry, in the fields of embedded system design and optimization, and security optimization of embedded systems and distributed embedded systems. In these fields, he co-authored of more than 70 publications on prestigious journals and conferences. His current research interests focus on proactive and reactive thermal-aware optimization of data centers, and design techniques and tools for energy-efficient compute-intensive embedded applications.

Kalyan Vaidyanathan is a Principal Engineer for Oracle and researcher with the System Dynamics Characterization and Control team in Oracle's Physical Sciences Research Center in San Diego. He received his M.S. and Ph.D. degrees in Electrical and Computer Engineering from Duke University.

Kenny Gross is a Distinguished Engineer for Oracle and researcher with the System Dynamics Characterization and Control team in Oracle's Physical Sciences Research Center in San Diego. He specializes in advanced pattern recognition, continuous system telemetry, and dynamic system characterization for improving the reliability, availability, and energy efficiency of enterprise computing systems, as well as the data centers in which the systems are deployed. He has 221 US patents issued and pending, 184 scientific publications, and was awarded a 1998 R&D 100 Award for one of the top 100 technological innovations of that year, for an advanced statistical pattern recognition technique that was originally developed for nuclear and aerospace applications and is now being used for a variety of applications to improve the quality-of-service, availability, and optimal energy efficiency for enterprise computer servers. He earned his Ph.D. in nuclear engineering from the University of Cincinnati.

Ayse K. Coskun (M'06) received the M.S. and Ph.D. degrees in Computer Science and Engineering from the University of California , San Diego. She is currently an Assistant Professor at the Department of Electrical and Computer Engineering, Boston University (BU), Boston, MA. She was with Sun Microsystems (now Oracle), San Diego, prior to her current position at BU. Her current research interests include energy-efficient computing, multicore systems, 3-D stack architectures, computer architecture, and embedded systems and software. Dr. Coskun received the Best Paper Award at IFIP/IEEE VLSI-SoC Conference in 2009 and in HPEC Workshop in 2011. She currently serves on the program committees of many design automation conferences including DAC, DATE, GLSVLSI, and VLSI-SoC. She has served as a guest editor in ACM TODAES journal and currently is an associate editor of IEEE Embedded Systems Letters. She is a member of the IEEE and ACM.